

WRDC-TR-90-8007
Volume VIII
Part 22

AD-A248 930



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 22 - Graph Language Development Specification

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209



September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

92-10065



MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

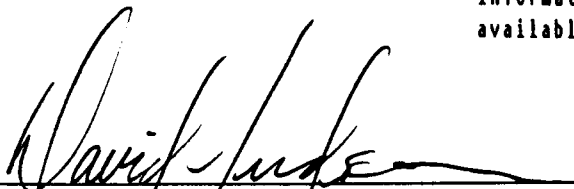
98 4 20 155

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.


This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations


DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:


BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS			
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE						
4. PERFORMING ORGANIZATION REPORT NUMBER(S) DS 620344403			5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC/MTI-TR- 90-8007 Vol. VIII, Part 22			
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services		6b. OFFICE SYMBOL (if applicable) WRDC/MTI		7a. NAME OF MONITORING ORGANIZATION WRDC/MTI		
6c. ADDRESS (City, State, and ZIP Code) 2970 Presidential Drive Fairborn, OH 45324-6209			7b. ADDRESS (City, State, and ZIP Code) WPAFB, OH 45433-6533			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF		8b. OFFICE SYMBOL (if applicable) WRDC/MTI		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464		
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6533			10. SOURCE OF FUNDING NOS.			
11. TITLE Graph Lr			See block 19		PROGRAM ELEMENT NO. 78011F	PROJECT NO. 595600
					TASK NO. F95600	WORK UNIT NO. 20950607
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S. , et al.						
13a. TYPE OF REPORT Final Report		13b. TIME COVERED 4 / 1 / 87 - 12 / 31 / 90		14. DATE OF REPORT (Yr., Mo., Day) 1990 September 30		15. PAGE COUNT 45
16. SUPPLEMENTARY NOTES WRDC/MTI Project Priority 6203						
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify block no.)			
FIELD	GROUP	SUB GR.				
1308	0905					
19. ABSTRACT (Continue on reverse if necessary and identify block number) This specification establishes the development, test, qualification, and performance requirements of an extension of the Forms Definition Language (FDL) identified as the Graph Definition Language (GDL). BLOCK 11: INTEGRATED INFORMATION SUPPORT SYSTEM Vol VIII -User Interface Subsystem Part 22						
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified			
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson			22b. TELEPHONE NO. (Include Area Code) (513) 255-7371		22c. OFFICE SYMBOL WRDC/MTI	

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

<u>SUBCONTRACTOR</u>	<u>ROLE</u>
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

	<u>Page</u>
SECTION 1.0 SCOPE	
1.1 Identification	1-1
1.2 Functional Summary	1-1
SECTION 2.0 DOCUMENTS	
2.1 Reference Documents	2-1
2.2 Terms and Abbreviations	2-1
SECTION 3.0 REQUIREMENTS	
3.1 Computer Program Definition	3-1
3.1.1 Interface Description	3-1
3.2 Detailed Functional Requirements	3-3
3.2.1 Graphical Display	3-3
3.2.1.1 Graph Definition	3-4
3.2.1.1.2 Additive versus Absolute Display	3-5
3.2.1.1.3 Attribute Definitions	3-5
3.2.1.1.4 Axis Information	3-5
3.2.1.1.5 Background Color	3-6
3.2.1.1.6 Color	3-6
3.2.1.1.7 Curve Information	3-6
3.2.1.1.8 Legends	3-7
3.2.1.1.9 Margins	3-7
3.2.1.1.10 Pie Chart Information	3-7
3.2.1.1.11 Text	3-8
3.3 Performance Requirements	3-8
3.3.1 Programming Methods	3-8
3.3.2 Program Organization	3-8
3.3.3 Expandability	3-8
3.3.4 Error Recovery	3-8
3.4 Data Base Requirements	3-8
3.5 Adaptation Requirements	3-9
SECTION 4.0 QUALITY ASSURANCE PROVISIONS	
4.1 Introduction and Definitions	4-1
4.2 Computer Programming Test and Evaluation	4-1
SECTION 5.0 PREPARATION FOR DELIVERY	5-1
SECTION 6.0 NOTES	6-1

APPENDICES

APPENDIX A GRAPH DEFINITION LANGUAGE	A-1
APPENDIX B HIGH-LEVEL INTERFACE	B-1
APPENDIX C INFORMATION MODEL	C-1

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
3-1	Compilation of Graph Definition	3-1
3-2	Application Environment on IISS	3-2
3-3	Data Transformation	3-2

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



SECTION 1

SCOPE

1.1 Identification

This specification establishes the development, test, qualification, and performance requirements of an extension of the Forms Definition Language (FDL) identified as the Graph Definition Language (GDL). This extension builds upon the graphics design work performed under contract as part of the Integrated Information Support System (IISS). In particular, it includes the 2-dimensional X-Y plot, pie chart, and bar chart definitions.

The GDL allows the creation of a dynamic graphical picture. The compilation of this language (with FLAN) produces a separate graphics object file for each graph.

1.2 Functional Summary

Graph definitions are comprised of two parts, much like form definitions. Graph fields must be referenced in a form definition to define their existence, size, and location. Then, the remaining general information about the graph and the particular data which will be displayed in that graph are defined in a CREATE GRAPH statement.

As is suggested by the form of the graph definition via a CREATE GRAPH statement, there is a certain hierarchical structure to the graph as there is with forms. The graphics field, like an item, is a terminus for the containing form. This implies that a graphics field may not contain non-graphics fields. A graph definition, however, may reference item fields via a path name or a list of path names. The data used to form the graph will then be taken from the items in sequence. If the list of path names is not present, a constant list of data may be supplied. If neither list is present, the graph is not displayed by the Form Processor.

SECTION 2

DOCUMENTS

2.1 Reference Documents

- [1] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620344200, 31 May 1988.
- [2] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620344400, 31 May 1988.
- [3] Structural Dynamics Research Corporation, C Coding Guidelines,
- [4] American National Standards Institute, Computer Graphics - Graphical Kernel System (GKS) Functional Description, ANSI X3.124-1985, 24 June 1985.
- [5] American National Standards Institute, Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information, ANSI X3.122-1986, 27 August 1986.
- [6] American National Standards Institute, Additional Controls for use with American National Standard Code for Information Interchange, ANSI X3.64-1975.

2.2 Terms and Abbreviations

Application Generator (AG): A subset of the IISS User Interface that consists of software modules that generate IISS application code and associated form definitions based on a language input. The part of the AG that generates report programs is called the Report Writer. The part of the AG that generates interactive applications is called the Rapid Application Generator.

Application Interface (AI): A subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process (AP): A cohesive unit of software that can be initiated as a unit to perform some function or functions.

Attribute: A field characteristic such as blinking, highlighted, black, etc., and various other combinations. Background attributes are defined for some forms or windows only. Foreground attributes are defined for items.

Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Closed Figure: A figure is closed if the path traced by a moving point returns to its starting position. The starting position may be arbitrarily assigned. "Fillarea" is synonymous with "closed figure".

Complex Figure: A figure is complex if the path traced by a moving point crosses itself. An arbitrary point may be determined to be contained within the traced boundary if a line drawn to infinity crosses the boundary an odd number of times. If the number of crossings is zero or even, the point is outside the traced boundary.

Dependent Data: Data correlated to a dependent variable.

Dependent Variable: A mathematical variable whose value is determined by that of one or more other variables in a function.

Device Drivers (DD): Software modules written to handle I/O for a specific kind of terminal. The modules map terminal-specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: An internal Form Processor list that contains only those forms that have been added to the screen and are currently displayed on the screen, along with information on where those forms are used.

Element: A graphics line or other primitive composed of graphics lines, such as an arc.

Field: In reference to the Forms Processor, "field" refers to any object on the open or display list. These objects can be forms, items, windows, etc.

In reference to graphs, "field" refers to a collection of one or more graph figures. A graph field can be an axis, curve, pie chart, grid, etc.

Figure: A collection of elements. A figure may be closed or open.

Fillarea: A collection of elements. A fillarea must be closed. "Closed figure" is synonymous with "fillarea".

Form: A structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, windows, prompts, non-graphics lines, and graphics.

Forms Definition Language (FDL): The language in which electronic forms are defined.

Forms Driven Form Editor (FD FE): A subset of the Form Editor which consists of a forms-driven application used to create and/or modify Form Definition files interactively.

Form Editor (FE): A subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor (FD FE) and the Forms Language Compiler (FLAN).

Form Hierarchy: A graphic representation of the way in which fields are related to their parent form.

Forms Language Compiler (FLAN): A subset of the Form Editor that consists of a batch process that accepts a series of Forms Definition Language (FDL) statements and produces form definition files as output.

Form Processor (FP): A subset of the IISS User Interface that consists of a set of callable execution-time routines available to an application program for form processing.

Graph: A picture correlated with data that alters as the data changes; by necessity, this is a dynamic (not pre-defined) picture. A graph may be imposed on windows or forms.

Graph Definition Language (GDL): An extension of the Forms Definition Language (FDL) which is used to define business graphs such as pie charts, X-Y plots, and bar charts.

Graph Figure: A collection of graphics primitives. The primitives can be circles, lines, arcs, etc.

Graphics Kernal System (GKS): A 2-dimensional graphics standard which is defined independently of any programming language.

Icon: A collection of figures and points that is pre-defined. An icon may be imposed on windows or forms. "Icon" is synonymous with "picture".

Independent Data: Data that is correlated to an independent variable.

Independent Variable: A mathematical variable whose value is specified first and determines the value of one or more other values in an expression or function. For example, in a business graph of sales versus month, month is the independent variable and sales is the dependent variable, because sales varies by month.

Integrated Information Support System: (IISS), a computing environment used to investigate, demonstrate, test the concepts and produce application for information management

and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: A non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined area where user data may be input/output.

Local Area Network (LAN): A privately owned network that offers reliable, high-speed communications channels optimized for connecting information processing equipment in a limited geographic area.

Message: Descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or to provide other user information.

Message Line: A line on the terminal screen that is used to display messages.

Open Figure: A figure is open if the path traced by a moving point does not return to its starting position. The starting position may be arbitrarily assigned. "Polyline" is synonymous with "open figure".

Open List: An internal Form Processor list that contains all forms that the application has opened for use along with information on where the form is used.

Operating System (OS): Software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: An instance of a form in a window that is created whenever a form is added to a window.

Physical Device: A hardware terminal.

Picture: A collection of figures and points that is pre-defined. A picture may be imposed on a window or a form. "Picture" is synonymous with "icon".

Picture Definition Language (PDL): An extension of the Forms Definition Language (FDL) which allows the definition of any graphics picture.

Point: A marker or a symbol.

Polyline: A collection of elements. A polyline must be an open figure. "Open figure" is synonymous with "polyline".

Primitive: The smallest unit of graphic detail. A graphic primitive can be a line, point, arc, etc.

Qualified Name: The name of a field preceded by the hierarchy path so that it is uniquely identified.

Report Writer (RW): Part of the Application Generator (AG) that generates source code for report programs based on language input.

Subform: A form that is used within another form.

Text Editor (TE): A subset of the IISS User Interface that consists of a file editor that is based on the text editing functions built into the Form Processor (FP).

User Data: Data which is either input by the user or output by the application programs to items.

User Interface (UI): A subsystem of IISS that controls the user's terminal and interfaces with the rest of the subsystem. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System (UIDS): A collection of IISS User Interface subsystems that is used by application programmers as they develop IISS applications. The UIDS includes the Form Editor (FE) and the Application Generator (AG).

User Interface Management System (UIMS): The run-time UI. It consists of the Form Processor (FP), Virtual Terminal (VT), Application Interface (AI), the User Interface Services (UIS), and the Text Editor (TE).

User Interface Services (UIS): A subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It includes message management, change password, and application definition services.

User Interface/Virtual Terminal Interface (UI/VTI): Another name for the User Interface.

Window: A dynamic area of a terminal screen on which pre-defined forms may be placed at run-time.

Window Manager: A facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor (FP).

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

The Graph Definition Language (GDL) is an extension to the Forms Definition Language (FDL) which will enable the definition of graphs through the User Interface software.

A graph may also be redefined dynamically through a series of dynamic graph routines.

3.1.1 Interface Description

This section and Appendix B describe the interfaces within the graphics software. In the following figures, GI stands for graphics interface, GKS is the graphics software which builds the internal data structures, GKS2 is the actual graphics software, and AI is the application interface. The AI routines are the calls to the UI which specify the actions to be performed. The FP, GI, GKS, and GKS2 systems perform the required actions. For graphics, clipping is performed in the VTI. Existing graphics applications written using the Fortran GKS binding will be able to call the comparable routines in the GKS subsystem. It should be noted that if the Graph Definition Language is used, the application will not normally be using the Graph or Picture AI routines.

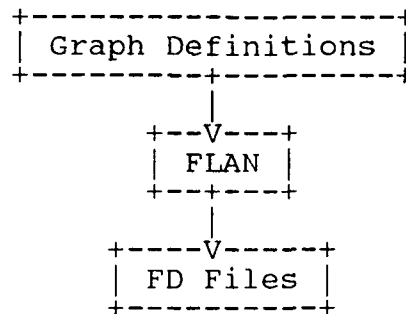


Figure 3-1 Compilation of Graph Definition

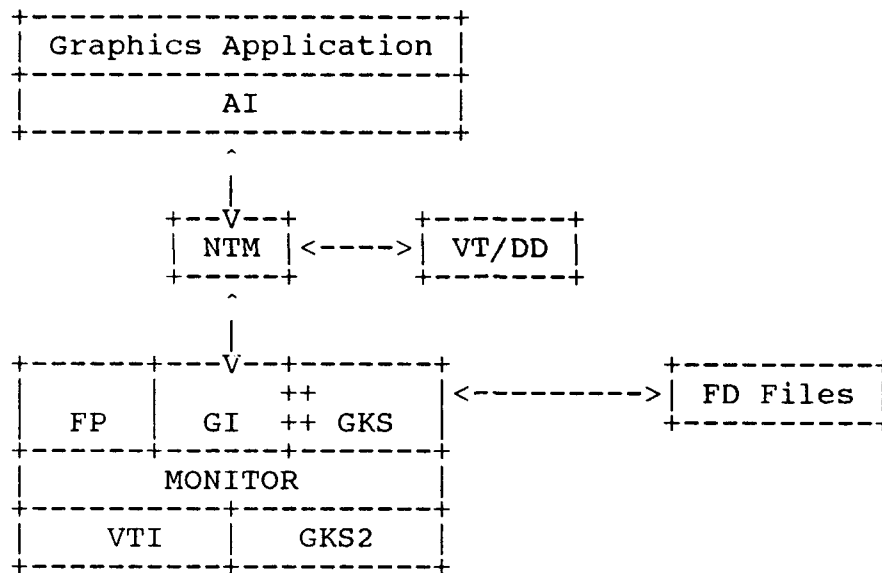


Figure 3-2 Application Environment in IISS

At present, there is no standard merging text and graphics. The existing standard will be enhanced to accommodate graphics. A lower-case v will be used to indicate graphics information. Individual primitives will be characterized by an additional intermediate character. The FP will integer normalize the world coordinate data to be in the range of 0 through 65535. The normalized integer data will be messaged to the Device Driver which will issue the commands to display it.

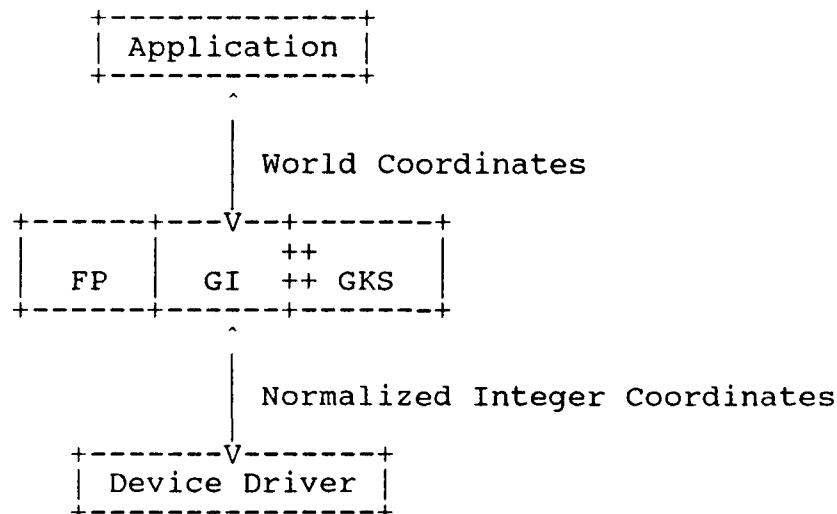


Figure 3-3 Data Transformation

<u>Primitive</u>	<u>Terminal Command</u>	<u>Arguments</u>	
graphics view	v	id, width, depth, row, column	
set graph view	=v	id	
delete view	<space>v	id	
erase view	\$v	id	
polyline	-v	attribute id, (x data, y data) ...	
polymarker	*v	attribute id, (x data, y data) ...	
fillarea	+v	attribute id, (x data, y data) ...	
text	"v	attribute id, upvector, path, horizontal alignment, vertical alignment, height, message string follows the terminal command	
attribute	!v	attribute id, type, then one of the following columns:	
<u>Polymarker</u>	<u>Polyline</u>	<u>Text</u>	<u>Fillarea</u>
marker type	line type	font	style
marker size	line width	gap	interior style
color	color	expansion	color
		color	

3.2 Detailed Functional Requirements

3.2.1 Graphical Display

The Rapid Application Generator (RAP) and the Form Processor (FP) will provide the capability to graphically display data in 2-dimensional format as X-Y plots, bar charts, or pie charts. The FP will provide this through the graphics field, which will contain a single graph of pre-defined type and structure. However, the associated graphics picture will be dynamic. In the case of X-Y plots and bar charts, more than one dataset may be plotted in a single graph to yield multiple curves or bars. An X-Y plot or bar chart may contain both additive and absolute dependent datasets for any given independent dataset. If both additive and absolute dependent data are present, the additive sets will be associated with the specified dependent dataset. The default dependent dataset to be added to is the first defined.

Size and location values are stated in terms of the default terminal character sizes and positions. No scaling is done automatically; the user has the responsibility of ensuring that information is of the proper size to fit in the available space. Portions of a picture which do not fit in a displayed space are clipped, not wrapped. The FP will attempt to adjust pictures appropriately for the aspect ratio of the device on which they are displayed.

Optional attributes will have default values set so that a graph can be included in a display with a minimum of effort at specification. Attributes unsupported on a given device (ex. color on a monochrome monitor) will default to appropriate values.

On devices supporting them, capabilities shall include a choice of line style and width (solid, dashed, dotted, dashed-dotted, and varying factors of thickness), and the choice of color for the curve, the background, and the area under the curve. Other available choices shall include the specification of the size and font style for text in axis labels, tick mark labels, legend entries, the lower and upper limits of axes, the scale (linear, logarithmic) of each axis, the presence or absence of a grid, the locations and lengths of the axes, some shading pattern below the curves, and symbols to appear at the data points. For graphs consisting of more than one curve there shall also be the option of displaying the dependent data additively or absolutely as measured on the axis.

Graphs shall be for output display only, with the exception that data points may be picked on the graph. The graphics picture will then be altered to retain only the picked area.

The data sources, if specified, for the graphs denoted by independent and dependent data in the language syntax must be either numeric item fields or constants. The item fields may or may not be displayed. If an item field is an input field, data in it can be altered so that the next time the graph which uses that field is displayed it will reflect the new value. It should be noted that if more than one value is associated with either an independent or a dependent dataset, and the source is an item field, the source may be defined as either a repeating numeric item field, or individual item fields. The repeating set of fields is preferred.

If an item will be receiving data from an NDML SELECT statement, then the form containing the item must be a repeating form field.

If no item fields are specified from which to obtain the data, the data must be listed in the USING and CURVE clauses of the graph definition. If more than one independent dataset is to be used, the independent dataset must be specified in the VERSUS clause of the CURVE definition. The USING clause indicates that the listed independent dataset is to be used for a curve that does not contain a VERSUS clause.

3.2.1.1 Graph Definition

The graph is defined as a graphics field on a form in the same way as other fields are defined. The graph field must also appear in a CREATE GRAPH statement. The size and location are required parts of the graph definition. The location of the graph can be defined relative to other fields in the form containing the graph. Prompts associated with the graph are external to the graph as they are for item fields.

3.2.1.2 Additive Versus Absolute Display

When data are displayed using X-Y plots, more than one dependent dataset may be plotted in a given graph. Two or more curves are distinguished by differing color and/or linestyle. Dependent values can be measured on the dependent axis, either relative to the independent axis or relative to a curve already displayed. The former method is absolute display, the latter is additive display. By default, displays will be absolute.

3.2.1.3 Attribute Definitions

Attribute bundles may be defined and named so that they can be easily referenced elsewhere in the graph definition. For text, attribute bundles may include the primitives FONT, COLOR, SIZE, and UPVECTOR. UPVECTOR specifies the up direction of the individual characters. The characters are placed along the line perpendicular to the character UPVECTOR. For curves, attribute bundles may include the primitives LINE_TYPE, LINE_WIDTH, COLOR, SYMBOL, and SYMBOL_FREQUENCY. SYMBOL_FREQUENCY instructs the FP to place a symbol at the first, last, and every nth datum position. The default frequency is one. If both LINE_TYPE and SYMBOL are specified, the data curve will be drawn and markers will then be placed over the curve at the indicated frequency. For curves, SIZE refers to the height of the marker.

3.2.1.4 Axis Information

Axis and tick mark labels can be specified separately for the dependent and independent axes. The text in these labels can be defined to have any available font, color, and size (in units of the terminal's standard character height and width). By default, text will have a size of 1 by 1 and a simple block font; the color will be the default contrasting color to the background (ex. white on black). The color of the axis itself including tick marks can also be specified and may differ from the color of the labels.

For a bar or an X-Y plot, if an axis is not specified, a default axis will be created based on the maximum and minimum data values associated with that axis.

The tick mark clause specifies the step between or the number of major tick marks and optionally the number of minor divisions between each pair of major tick marks. The tick mark label is comprised of a sequence of strings. If the number of strings is less than the number of major tick marks, then the tick mark labels are reused from the beginning. If the tick mark label is not specified and the maximum and minimum axis values are numeric, the labels will be generated.

If no label string is defined for an axis, the associated item field name is used as the label. In a case where more than one item field is associated with a vertical axis, the name of the first such item field is used. If no tick marks are

defined, then the axis is divided into a number of intervals which yield nice tick mark label values. Nice values are defined to be multiples by powers of ten of values in the set {.1, .2, .25, and .5}.

The lower and upper limits and the scale (linear or logarithmic) of the axes may also be specified independently. If the range of the data exceeds the limits specified, then data outside of the range are not displayed. Unspecified limits are set by default to give the smallest range which will span the entire range of the data such that the tick mark labels can be either those given by the user or those in the set of nice values as defined in the paragraph above. If the scale is not defined to be logarithmic, it is linear by default.

X-Y plots and bar charts can be defined to have grids superimposed upon them. The default choice is no grid displayed. The grid is comprised of lines of constant value of the coordinate specified. If minor tick marks have been specified, then a fine grid will connect to minor tick marks. If minor tick marks have been defined for only one axis, then fine grid lines will appear perpendicular only to that axis.

3.2.1.5 Background Color

The background of the graph field can be defined to have any available color.

3.2.1.6 Color

On terminals supporting color, the following colors are defined: red, yellow, green, cyan, magenta, blue, white, and black. Unless the defaults are used, it is the user's responsibility to ensure visibility of the graph components by selecting contrasting colors. Improper choice of colors may render some components invisible. For example, defining the background as white and using white to draw the axes and tick marks will make the axes indistinguishable from the background.

3.2.1.7 Curve Information

For X-Y plots, several attributes of the curves may be specified, including the color of the curve as well as the color or shading pattern of the area below the curve. On devices supporting the feature, the line type (solid, dashed, etc.) and line width may also be specified. It is possible to specify an additional optional pattern that is to be used if the physical device is monochromatic using the OR clause of the SHAPE or DISPLAY AS specification. For X-Y plots, symbols may optionally be placed at data point locations using a specified frequency, either in addition to or instead of the curve. By default, data points are connected by solid line segments in the order of occurrence of the source item field values, and no data point symbols appear. The symbols may be chosen from a catalog of pre-defined symbols and will include minimally the dot, plus sign, asterisk, cross or x, and circle. By default, no shading occurs below curves and the color is the background color.

3.2.1.8 Legends

An X-Y plot, bar chart, or pie chart can be defined to have a legend in a specified location. By default, no legend will appear. The legend can optionally be circumscribed by a box. The location value is the location of the upper left-hand corner of the legend, relative to the graph. The legend entry is defined in the CURVE clause or the PIE clause of the CREATE GRAPH statement. The attribute name associated with the legend may be used to specify the font, size, and color attributes for the legend entry text.

3.2.1.9 Margins

The amount of space surrounding the graph which is delimited by the axes is adjustable. This space can be set by the location specifications for the axes in the CREATE GRAPH statement. By default, all margins are 10 percent.

3.2.1.10 Pie Chart Information

The data source for a pie chart, if given, may be a single repeating item field or list of item fields. If the repeat factor is one or the list consists of one item field, then the pie chart consists of one circular segment. The segments correspond in order to the elements of the item field. If the data source is not indicated, a constant list may be given.

Some pie chart attributes are associated with individual segments. The segments are numbered sequentially in counter-clockwise fashion from one beginning at a horizontal radius vector to the right.

The explosion factor, which is applied on a segment by segment basis, is the percentage of the radius of the pie by which a segment is projected radially outward. By default, no segments are exploded.

Individual segments may be labeled with a string of specified font, size, and color. The labels may be placed wither inside or outside of the segment. Alternatively, legend entries may be defined for each segment.

The percentages of the whole may be displayed for each segment, either inside or outside of the segment.

The item field values may be displayed for each segment, either inside or outside of the segment, by requesting the quantity.

Each segment may be individually colored or shaded with a pattern. By default, no pattern will appear with the segments and the interior color will be the background color of the graph field. It is possible to specify an additional optional pattern that is to be used if the physical device is monochromatic using the OR clause of the SHADE specification.

3.2.1.11 Text

Text may be included anywhere within the graph field for titles, annotation, etc., via the LABEL clause. The GDL LABEL clause is similar to the FDL PROMPT clause with the exception that the text attributes include FONT, SIZE, and, if supported by the device, COLOR and UPVECTOR. UPVECTOR specifies the up direction of the individual characters. The characters are placed along the line perpendicular to the character UPVECTOR.

3.3 Performance Requirements

3.3.1 Programming Methods

The Business Graph capability of the Graphics Subsystem will be provided as an extension to the Forms Definition Language (FDL) and the Rapid Application Generator (RAP). In addition, a set of callable services which will be accessible from within an application program will be provided to create, modify, and delete a business graph.

3.3.2 Program Organization

The Graphics Subsystem will be developed in a modular way and will comprise functionally cohesive modules. Along with being a new subsystem, it will be configured as an extension of the Form Processor (FP) and Virtual Terminal Interface (VTI) subsystems of the User Interface.

3.3.3 Expandability

The Business Graph capability of the Graphics subsystem will be developed with a view to admitting future possible extensions in terms, perhaps, of needle graphs and 3-dimensional business graphs. Also, to the extent feasible, constraints which are imposed in the current implementation will not be incorporated as fundamental and mandatory design limitations but as restrictions which may be relaxed if warranted in the future.

3.3.4 Error Recovery

The Graphics Subsystem of the UI will recover from errors which may arise in its use and return appropriate error codes to indicate the nature and cause of the error. In addition, messages will be displayed to the user in the message line.

3.4 Data Base Requirements

Additional Form Processor data structures will be needed to contain the graph definition information used by the graphics software to construct the graph pictures. By keeping this information, it will be possible for the window manager to move or overlay graphs along with graphics pictures.

The following structures represent the type of information that is necessary for a general graphics capability. The individual names may change when implemented. For example, "bundle" is similar to the current FP attribute, and the name is used merely to point out that we are referring to a GRAPHICS attribute at this point.

For graphics pictures and graphs, data structures are created and initially populated by FLAN and stored in graphics definition object files during the compilation of the low-level PDL or high-level GDL.

At the time the form containing the graph is output, the data contained in the graphics data structures are transferred to the graphics software. If the graph attributes are to be changed at run-time, the application must issue the appropriate dynamic graph routines.

3.5 Adaptation Requirements

The Graphics Subsystem of the UI will be required to be completely integrated with the rest of the UI. It will, therefore, be required to operate in those environments hosting IISS. No modifications to applications shall be necessary in order to rehost to a different host processor provided the published coding guidelines are followed.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definitions

"Testing" is a systematic process that may be pre-planned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur, and, when they do occur, to make them more noticeable to the programmer and the user. In other words, as much error checking as is practical and possible in each routine should be performed.

4.2 Computer Programming Test and Evaluation

The quality assurance provisions for testing consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests are performed by the design team. Structured design, design walk-through, and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

Each function is tested separately, then the entire subsystem is tested as a unit. Development and testing are done on the Air Force VAX.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the Integrated Information Support System (IISS) Test Bed site located at Arizona State University, Tempe, Arizona. The software associated with each CPC released is delivered on a media which is compatible with the IISS Test Bed. The release is clearly identified and includes procedures to be followed for installation of the release. Integration with the other IISS CPCs is done on the IISS Test Bed on a scheduled basis.

SECTION 6

NOTES

Please refer to the Software Availability Bulletin, Volume III, Part 16, CI# SAB620326000, for current IISS software and documentation availability.

APPENDIX A

GRAPH DEFINITION LANGUAGE

This appendix contains the Graph Definition Language in its entirety. It should be noted that the low-level Graphics or Picture Definition Language is not addressed in this document. The following notation is used to describe the syntax of the GDL.

UPPER-CASE	Identifies reserved words that have specific meanings in the GDL. These words are generally required unless the portion of the statement containing them is itself optional.
lower-case	Identifies names, numbers, or character strings that the user must supply.
Initial upper-case	Identifies a statement or parameter that is defined later on.
_ Underscore	Identifies reserved words or portions of reserved words that are optional.
{ } Braces	Enclosing vertically stacked options indicate that one of the enclosed options is required.
[] Brackets	Indicate that the enclosed statement or parameter is optional.
... Ellipsis	Indicates that the preceding statement or parameter may be repeated any number of times.

Field Definition - Graphs

```
GRAPH graph_name
  location
  SIZE cols BY rows
  PROMPT location "string"
```

Graph Definition

```
CREATE { BAR } GRAPH graph_name
      { PIE }
      { LINE }

[ USING ( [ independent data ] [ AXIS axis_name ] ) ]
[ LEGEND Location [ BOX ] ]

+-- LABEL { Location, [ DISPLAY AS attribute_name ] }
|         { DISPLAY AS attribute_name, Location }
|         "string"
+-- ...
```

```
[ ATTRIBUTE attribute_name  
    ( Primitive [ ,Primitive ] ... ) ] ...
```

```
+--                                     +-  
  AXIS axis_name  
    DISPLAY AS attribute_name  
  
    { HORIZONTAL }  
    { VERTICAL   }  
  
    [ LABEL attribute_name "string" ] ...  
    [ Location ]  
    [ MINIMUM lower_limit ]  
    [ MAXIMUM upper_limit ]  
  
    +-                               +-  
    | SCALE { LINEAR } |  
    |           { LOG  } |  
    +-                               +-  
  
    [ SIZE length ]  
    [ TICK [ EVERY ] ndiv [ minor ] attribute_name  
      "string" ... ]  
  
    [ [ FINE ] GRID ]  
+--                                     +- ...
```

```
[ BACKGROUND attribute_name ]
```

```
+--                                     +-  
  CURVE curve_name  
    dependent data [ USING AXIS axis_name]  
                  [ VERSUS independent data ]  
    [ DISPLAY AS attribute_name  
      [ OR attribute_name ] ]  
    [ LEGEND attribute_name "string" ]  
  
    [ SHADE [ COLOR color ] [ PATTERN pattern ]  
          [ OR pattern ] ]  
  
    +-                               +-  
    | { ABSOLUTE } |  
    | { ADDITIVE  } [ USING CURVE curve_name ] |  
    +-                               +-  
+--                                     +- ...
```

```

+-
|  PIE segment_number
|  [ EXPLODE fraction ]
|
|  [ LABEL attribute_name "string" ]
|  [ LEGEND attribute_name "string" ]
|
|  +-
|  | PERCENT { INSIDE } attribute_name +-
|  |           { OUTSIDE }
|  +-
|
|  +-
|  | QUANTITY { INSIDE } attribute_name +-
|  |           { OUTSIDE }
|  +-
|
|  [ SHADE [ COLOR color ] [ PATTERN pattern ]
|           [ OR pattern ] ]
+-
|
+- ...

```

Primitives

```

[ DISPLAY      color ]
[ FONT         font ]
[ LINE_TYPE    line_type ]
[ LINE_WIDTH   line_width ]
[ SIZE         size ]
[ SYMBOL       symbol_id [ [ SYMBOL_FREQUENCY ] frequency ] ]
[ UPVECTOR     angle ]

```

APPENDIX B

HIGH-LEVEL INTERFACE

This appendix contains the high-level routines for the Business Graph capability of the Graphics subsystem. These routines are used by the application programmer to dynamically create or alter a graph definition. If the displayed graph alters only when the data used to draw the graph changes, these routines need not be used if the "where data is located" clause is used.

The routines that affect the display list only are, for the most part, data-dependent. Because of this, it is possible to use the same graph definition on multiple forms, yet have a different graph in each instance.

DEFSEG - Used to create or redefine a segment of a pie chart. This routine affects the display list only.

inputs: path_name, character, 120 bytes, the
unique path to the graph

segment_number, integer, identifies the
segment

explosion_factor, double precision,
percentage of the radius to project the
segment from the center of the circle.
For example, an explosion factor of 2
will project the segment a factor of
(radius + (0.02 * radius)) from the
circle center.

fill_color, character, 7 bytes, the
color with which to fill the segment.
The available colors are BLACK, WHITE,
MAGENTA, BLUE, RED, CYAN, YELLOW, GREEN.

pattern, character, 10 bytes, name of
the pattern with which to fill the
segment.

alt_pattern, character, 10 bytes, name
of the pattern to use if the physical
device is monochromatic

outputs: rcode, character, 5 bytes, status
completion code

PERQUA - Used to add a percent or quantity definition to a segment. This routine affects the display list only.

inputs: path_name, character, 120 bytes, the
 unique path identifier to the graph

 segment_number, integer, the segment
 identifier

 perquan, character, 1 byte, (P)ercent
 or (Q)uantity

 inout, character, 1 byte, (I)nside or
 (O)utside

 attrib, character, 10 bytes, the name
 of the graphics attribute bundle to use

outputs: rcode, character, 5 bytes

LABSEG - Used to add a label to a pie segment. A segment may have more than one label by calling this routine the desired number of times. The routine affects the display list only.

inputs: path_name, character, 120 bytes, the
 unique path identifier to the graph

 segment_number, integer, identifies the
 segment

 length, integer, length of the segment
 label

 label, character, text which the segment
 is to be labeled with

outputs: rcode, character, 5 bytes

LEGLAB - Used to add a legend label for a curve or segment. Any number of legend labels may be added for a entity by calling the routine repeatedly. The routine affects the display list only.

inputs: path_name, character, 120 bytes, the
 unique path identifier to the graph

 leg_type, character, 1 byte, (C)urve or
 (S)egment

 segment_number, integer, used to
 identify the segment. Used only if
 leg_type is S.

cur_name, character, 10 bytes, used to identify the curve. Used only if leg_type is C.

length, integer, length of the legend string

legend, character, the legend string to be displayed

attrib, character, 10 bytes, the name of the graphics attribute bundle that is to be used to display the legend string

outputs: rcode, character, 5 bytes

DELSEG - Used to delete a pie segment. The data will also be deleted, and this will affect the drawing of the other segments in a pie chart since the total value of the pie will be altered. This routine affects the display list only.

inputs: path_name, character, 120 bytes, the unique path identifier to the graph

segment_number, integer, identifies the segment to delete

outputs: rcode, character, 5 bytes

DEFGRA - Used to define the graph. This routine affects the open and display lists.

inputs: graph_name, character, 10 bytes, name of the graph

type, character, 1 bytes, (P)ie chart, (C)urve plot, or (B)ar chart

attrib, character, 10 bytes, the name of the attrib bundle that is to be used in the background of the graph

outputs: rcode, character, 10 bytes

GRALOC - Used to specify where the graph is to be located on the containing form. This routine affects the display list only.

inputs: path_name, character, 120 bytes, the
unique path identifier to the graph

vfldnam, character, 120 bytes, the path
name of the vertical reference

vexref, integer, vertical point on the
reference field (1=top, 2=center,
3=bottom)

vinref, integer, vertical reference
point on the graph (1=top, 2=center,
3=bottom)

vofset, integer, vertical distance from
reference field to graph reference point

hfldnam, character, 120 bytes, path name
of the horizontal reference

hexref, integer, horizontal point on the
reference field (1=top, 2=center,
3=bottom)

hinref, integer, horizontal reference
point on the graph (1=top, 2=center,
3=bottom)

hofset, integer, horizontal distance
from reference field to graph reference
point

outputs: rcode, character, 5 bytes

DELGRA - Used to delete a graph definition from both the
open and display lists.

inputs: graph_name, character, 10 bytes, the
name of the graph

outputs: rcode, character, 5 bytes

ADDLEG - Used to add a legend definition to a graph. The
legend labels are added separately. The routine
affects the display list only.

inputs: path_name, character, 120 bytes, the
unique path identifier for the graph

enclose, character, 1 byte, (Y)es or
(N)o. This argument indicates whether
the legend is to be circumscribed by a
box.

vfldnam, character, 120 bytes, the path name of the vertical reference

vexref, integer, vertical point on the reference field (1=top, 2=center, 3=bottom)

vinref, integer, vertical reference point on the graph (1=top, 2=center, 3=bottom)

vofset, integer, vertical distance from reference field to graph reference point

hfldnam, character, 120 bytes, the path name of the horizontal reference

hexref, integer, horizontal point on the reference field (1=top, 2=center, 3=bottom)

hinref, integer, horizontal reference point on the graph (1=top, 2=center, 3=bottom)

hofset, integer, horizontal distance from reference field to graph reference point

outputs: rcode, character, 5 bytes

ADGLAB - Used to add a label to a graph. A graph may have any number of labels by calling the routine repeatedly. The routine effects the open and display list.

inputs: graph_name, character, 10 bytes, the name of the graph

length, integer, the length of the label string

label, character, the label string

attrib, character, 10 bytes, the name of the graphics attribute bundle used to display the label

vfldnam, character, 120 bytes, the path name of the vertical reference

vexref, integer, vertical point on the reference field (1=top, 2=center, 3=bottom)

vinref, integer, vertical reference
point on the graph (1=top, 2=center,
3=bottom)

vofset, integer, vertical distance from
reference field to graph reference point

hfldnam, character, 120 bytes, the path
name of the horizontal reference

hexref, integer, horizontal point on the
reference field (1=top, 2=center,
3=bottom)

hinref, integer, horizontal reference
point on the graph (1=top, 2=center,
3=bottom)

hofset, integer, horizontal distance
from reference field to graph reference
point

outputs: rcode, character, 5 bytes

DELLEG - Used to delete a legend from a graph. The routine
effects the display list only.

inputs: path_name, character, 120 bytes, the
unique path identifier to the graph

outputs: rcode, character, 5 bytes

DELGLA - Used to remove all the graph labels. The labels
on the axes or segments are not effected. This
routine effects the display and the open lists.

inputs: graph_name, character, 10 bytes, the
name of the graph

outputs: rcode, character, 5 bytes

DEFRCRV - Used to define a curve for a bar or linear plot.
This routine effects the display list only.

inputs: path_name, character, 120 bytes, the
unique path identifier to the graph

cur_nam, character, 10 bytes, the name
of the curve

axis, character, 10 bytes, the name of
the dependent axis

dsp_attrb, character, 10 bytes, the name of the graphics attribute bundle that the curve itself is to be displayed with

dsp_alt, character, 10 bytes, the name of the alternate graphics bundle that is to be used for display when the physical device is monochromatic

fill_color, character, 7 bytes, the name of the fill color. The available colors are BLACK, WHITE, MAGENTA, YELLOW, RED, BLUE, CYAN, GREEN.

fill_patt, character, 10 bytes, the name of the pattern that is to be used to fill the area under the curve

fill_alt, character, 10 bytes, the name of the alternate fill pattern to use if the physical device is monochromatic

type, character, 1 byte, (P)lus (additive) or (A)bsolute

add_name, character, 10 bytes, the name of the curve to be added to. This argument is used only when type is P.

outputs: rcode, character, 5 bytes

DELCRV - Used to delete a curve from the display list.

inputs: path_name, character, 10 bytes, the unique path identifier to the curve

outputs: rcode, character, 5 bytes

DELWHR - Used to delete a "where data is located" expression on a business graph. The routine effects the display list only.

inputs: path_name, character, 120 bytes, the unique path identifier to the graph (for main independent dataset) or the curve (for dependent dataset or specific independent dataset)

outputs: rcode, character, 5 bytes

ADDWHR - Used to add a "where data is located" expression to a dataset. If the clause is to be a list, the routine can be called more than once. Only the display list is effected by the routine.

inputs: path_name, character, 120 bytes, the unique path identifier to the graph (for main independent dataset) or the curve (for dependent dataset or specific independent dataset)

where_clause, character, 120 bytes, the path name of the location of the data

type, character, 1 byte, (I)ndependent (for bar graph and linear plot, this generates the VERSUS clause) or (D)ependent (do not use D with pie charts)

outputs: rcode, character, 5 bytes

ADDIAX - Used to specify the axis that is to be used for the independent axis. If the graph is a pie chart, an error is returned. Only the display list is effected by the routine.

inputs: path_name, character, 120 bytes, the unique path identifier to the graph

axis, character, 10 bytes, the name of the axis to use as the independent axis

outputs: rcode, character, 5 bytes

PGDATA - Used to provide data to the Form Processor in lieu of a "where data is located" clause. If the data location clause is already present, an error is returned and the clause must be removed.

inputs: path_name, character, 120 bytes, the unique path identifier to the graph (for main independent dataset) or the curve (for dependent dataset or specific independent dataset)

number, integer, the number of data points that are being passed in by the following argument

data, array, double precision, the data that is to be plotted

type, character, 1 byte, (I)ndependent
(for bar graph and linear plot, this
generates the VERSUS clause) or
(D)ependent (do not use D with pie
charts)

outputs: rcode, character, 5 bytes

DELBUN - Used to delete a graphics attribute bundle. If
the name of the bundle is omitted, all bundled
attributes are deleted. This routine effects both
the open and display lists.

inputs: graph_name, character, 10 bytes, name of
the graphics field. This is either the
name of the graph or the picture.

bundle_name, character, 10 bytes, the
name of the graphics attribute bundle

outputs: rcode, character, 5 bytes

DEFBUN - Used to define a graphics attribute bundle. This
routine effects the open and display lists.

inputs: graph_name, character, 10 bytes, the
name of the graph that the bundle is
associated with

bundle_name, character, 10 bytes, the
name of the graphics attribute bundle

length, integer, the length of the
attribute string

attrib, character, list of attributes
for the bundle being defined. The
individual attributes are separated by
spaces and/or commas.

outputs: rcode, character, 5 bytes

DELTLA - Used to delete the tick mark labels from a
particular axis instance on the display list.

inputs: path_name, character, 120 bytes, the
unique path identifier to the axis

type, character, 1 byte, (M)ajor, (S)
minor tick labels, or (B)oth major and
minor

- outputs: rcode, character, 5 bytes
- DELSLA - Used to delete the labels from a pie segment.
This routine effects the display list only.
- inputs: path_name, character, 120 bytes, the
unique path identifier to the axis
- segment_number, integer, used to
identify the segment
- outputs: rcode, character, 5 bytes
- DELALA - Used to delete all the labels from an axis
instance on the display list.
- inputs: path_name, character, 120 bytes, the
unique path identifier for the axis
- outputs: rcode, character, 5 bytes
- DELAXS - Used to delete an axis definition from both the
open and display lists.
- inputs: graph_name, character, 10 bytes, name of
the graph
- axis, character, 10 bytes, axis name
- outputs: rcode, character, 5 bytes
- DFMXMN - Used to define the maximum and minimum values on
an axis instance on the display list. If the
values are set, the FP will determine the
appropriate values based on the data.
- inputs: path_name, character, 120 bytes, the
unique path identifier to the axis
- max, double precision, the maximum value
on the axis
- min, double precision, the minimum
value on the axis
- outputs: rcode, character, 5 bytes
- DFAXLC - Used to determine where the axis is to be placed.
This routine effects the display list only.
- inputs: path_name, character, 120 bytes, the
unique path identifier to the axis
- vfldnam, character, 120 bytes, the path
name of the vertical reference

vexref, integer, vertical point on the reference field (1=top, 2=center, 3=bottom)

vinref, integer, vertical reference point on the graph (1=top, 2=center, 3=bottom)

vofset, integer, vertical distance from reference field to graph reference point

hfldnam, character, 120 bytes, the path name of the horizontal reference

hexref, integer, horizontal point on the reference field (1=top, 2=center, 3=bottom)

hinref, integer, horizontal reference point on the graph (1=top, 2=center, 3=bottom)

hofset, integer, horizontal distance from reference field to graph reference point

outputs: rcode, character, 5 bytes

ADDTLA - Used to add tick mark labels to an axis instance on the display list. Multiple labels may be added by repeating the routine call.

inputs: path_name, character, 120 bytes, the unique path identifier to the axis

type, character, 1 byte, (M)ajor or (S) minor

length, integer, the length of the tick label string

label, character, the tick label string

outputs: rcode, character, 5 bytes

ADDALA - Used to add axis labels to an axis instance on the display list. Multiple labels may be added by repeating the routine call.

inputs: path_name, character, 120 bytes, the unique path identifier to the axis

length, integer, the length of the axis string

label, character, the axis string

outputs: rcode, character, 5 bytes

DEFWIN - Used to define a graphics window. The limits of the window allow the data coordinates to be correlated to the screen coordinates. If the maximum and minimum coordinates of the window are not specified, the default for business graphs will be the maximum and minimum data values adjusted for a 10% margin. Only the display list is effected by this routine.

inputs: path_name, character, 120 bytes, the
unique path identifier to the graph

hmax, double precision, the maximum
horizontal value

hmin, double precision, the minimum
horizontal value

vmax, double precision, the maximum
vertical value

vmin, double precision, the minimum
vertical value

outputs: rcode, character, 5 bytes

DEFGVU - Used to define the extent that the graph is to fill the graphics window (the range is 0 to 1). Only the display list is effected by this routine.

inputs: path_name, character, 120 bytes, the
unique path identifier to the graph

hmax, double precision, the maximum
horizontal value

hmin, double precision, the minimum
horizontal value

vmax, double precision, the maximum
vertical value

vmin, double precision, the minimum
vertical value

outputs: rcode, character, 5 bytes

DEFAXS - Used to define an axis. Both the display and open lists are effected.

inputs: graph_name, character, 10 bytes, the name of the graph the axis is linked to

axis_name, character, 10 bytes, axis name

direc, character, 1 byte, (V)ertical or (H)orizontal

scale, integer, 0 - linear or 1 - logarithmic

grid, integer, 0 - none, 1 - major tick marks, 2 - minor tick marks, or 3 - both major and minor tick marks

length, integer, the length of the axis. A zero indicates that the FP is to determine the length

attrib, character, 10 bytes, name of the graphics attribute bundle to be used for display purposes

ADTICS - Used to add tick marks to a particular axis instance on the display list

inputs: path_name, character, 120 bytes, the unique path identifier to the axis

attrib, character, 10 bytes, the name of the graphics attribute bundle to be used to display the tick mark labels

type, character, 1 byte, (M)ajor or (S) minor

step, integer, 0 - number indicates step, 1 - number indicates absolute number of tick marks

number, double, either the step or the number of marks (if the number is not integral, an error is indicated) depending on step

outputs: rcode, character, 5 bytes

APPENDIX C
INFORMATION MODEL

